

Claims

WHAT IS CLAIMED IS:

1. A method comprising:
receiving into an execution environment input component code and a runtime security policy; and
generating a call graph of call paths through the input component code simulated in combination with at least one symbolic component representing additional arbitrary code that complies with the runtime security policy.
2. The method of claim 1 wherein a possible execution path through the input component code that is compliant with the runtime security policy is represented by an individual call path.
3. The method of claim 1 wherein at least one node in the call graph includes a symbolic permission set and a known method implementation.
4. The method of claim 1 wherein at least one node in the call graph includes a symbolic permission set and a token representing an unknown method implementation.
5. The method of claim 1 wherein the generating operation comprises:
initializing a symbolic value that represents data values that may be obtained by the arbitrary code at runtime.

1 6. The method of claim 1 wherein the generating operation comprises:
2 updating a symbolic value that represents data values that may be obtained
3 by the arbitrary code at runtime based on detection of an additional data value that
4 may be passed as a parameter to the arbitrary code at runtime.

5 7. The method of claim 1 wherein the generating operation comprises:
6 updating a symbolic value that represents data values that may be obtained
7 by the arbitrary code at runtime based on detection of a new dataflow to the
8 arbitrary code.

9 8. The method of claim 1 wherein the generating operation comprises:
10 generating a class hierarchy that contains classes of the input component
11 code and symbolic classes that represent classes of arbitrary code.
12

13 9. The method of claim 1 wherein the generating operation comprises:
14 identifying one or more methods of the input code component that can be
15 called by the arbitrary code.
16

17 10. The method of claim 1 wherein the generating operation comprises:
18 identifying one or more methods of the input component code that can be
19 called by the arbitrary code; and
20 identifying one or more other methods of the input component code that
21 can be called by the identified one or more methods of the input component code.

22 11. The method of claim 1 wherein the generating operation comprises:
23 identifying one or more methods of the input component code that can be
24 called by the arbitrary code; and
25

1 identifying at least one method of the arbitrary code that can be called by a
2 virtual call of the identified one or more methods of the input component code.

3 12. The method of claim 1 wherein any method reachable by execution in
4 accordance with the runtime security policy is represented by one of more nodes
5 in the call graph.

6
7 13. The method of claim 1 wherein the generating operation comprises:
8 generating at least one constraint associated with one or more instructions
9 in the input component code.

10 14. The method of claim 1 wherein the generating operation comprises:
11 generating at least one constraint associated with a parameter of a method
12 call in the input component code.

13
14 15. The method of claim 1 wherein the generating operation comprises:
15 generating at least one constraint associated with a returned result of a
16 method call in the input component code.

17 16. The method of claim 1 further comprising:
18 analyzing the call graph to identify a call path that presents a security
19 vulnerability in the input component code.

20
21 17. The method of claim 1 further comprising:
22 analyzing the call graph to identify a call path that presents a security
23 vulnerability in the input component code and a call path that presents no security
24 vulnerability in the input component code.
25

1 18. The method of claim 1 further comprising:
2 analyzing the call graph to generate a security report that identifies a
3 security vulnerability in the input component code.

4 19. The method of claim 1 further comprising:
5 analyzing the call graph to identify a call path that satisfies a query.
6

7 20. The method of claim 1 further comprising:
8 analyzing the call graph to identify a security-vulnerable usage of a
9 permission demand.

10 21. The method of claim 1 further comprising:
11 analyzing the call graph to identify a security-vulnerable usage of a
12 permission assertion.
13

14 22. The method of claim 1 further comprising:
15 analyzing the call graph to identify a lack of uniform usage of security
16 checks.

17 23. The method of claim 1 further comprising:
18 analyzing the call graph to identify an equivalence between use of a
19 permission link-demand and a permission demand.
20

21 24. The method of claim 1 wherein the generating operation comprises:
22 generating a class hierarchy that contains classes of the input component
23 code and symbolic classes that represent classes of the arbitrary code;
24 generating at least one constraint associated with a virtual call in the input
25 component code; and

1 evaluating the at least one constraint by a symbolic computation on
2 potential target classes for the virtual call in the generated class hierarchy.

3 25. The method of claim 1 wherein the generating operation comprises:
4 generating at least one constraint associated with either a security demand
5 or a security assert in the input component code;

6 evaluating the at least one constraint by a symbolic computation on
7 dynamic permissions of the input component code and on a parameter permission
8 of the security demand or the security assert; and

9 conditionally generating at least one additional constraint associated with
10 one or more instructions located in the input component code after the security
11 demand or assert, responsive to the evaluating operation.

12
13 26. The method of claim 1, further comprising
14 analyzing the call graph to classify, based on permissions, pieces of code
15 performing sensitive actions in the input component code.

16 27. The method of claim 1, further comprising
17 analyzing the call graph and another call graph obtained for a different
18 version of input component code to generate a security report that identifies a
19 security vulnerability in the different version of the input component code.

20
21 28. The method of claim 1, further comprising
22 analyzing the call graph and another call graph obtained for a different
23 version of input component code to identify a call path that presents a security
24 vulnerability in the different version of the input component code.

1 29. A computer program product encoding a computer program for
2 executing on a computer system a computer process, the computer process
3 comprising:

4 receiving into an execution environment input component code and a
5 runtime security policy; and

6 generating a call graph of call paths through the input component code
7 simulated in combination with at least one symbolic component representing
8 additional arbitrary code that complies with the runtime security policy.

9
10 30. The computer program product of claim 29 wherein a possible
11 execution path through the input component code that is compliant with the
12 runtime security policy is represented by an individual call path.

13 31. The computer program product of claim 29 wherein at least one node in
14 the call graph includes a symbolic permission set and a known method
15 implementation.

16
17 32. The computer program product of claim 29 wherein at least one node in
18 the call graph includes a symbolic permission set and a token representing an
19 unknown method implementation.

20 33. The computer program product of claim 29 wherein the generating
21 operation comprises:

22 initializing a symbolic value that represents data values that may be
23 obtained by the arbitrary code at runtime.

1 34. The computer program product of claim 29 wherein the generating
2 operation comprises:

3 updating a symbolic value that represents data values that may be obtained
4 by the arbitrary code at runtime based on detection of an additional data value that
5 may be passed as a parameter to the arbitrary code at runtime.

6 35. The computer program product of claim 29 wherein the generating
7 operation comprises:

8 updating a symbolic value that represents data values that may be obtained
9 by the arbitrary code at runtime based on detection of a new dataflow to the
10 arbitrary code.

11 36. The computer program product of claim 29 wherein the generating
12 operation comprises:

13 generating a class hierarchy that contains classes of the input component
14 code and symbolic classes that represent classes of arbitrary code.

15 37. The computer program product of claim 29 wherein the generating
16 operation comprises:

17 identifying one or more methods of the input component code that can be
18 called by the arbitrary code.

19 38. The computer program product of claim 29 wherein the generating
20 operation comprises:

21 identifying one or more methods of the input component code that can be
22 called by the arbitrary code; and
23
24
25

1 identifying one or more other methods of the input component code that
2 can be called by the identified one or more methods of the input component code.

3 39. The computer program product of claim 29 wherein the generating
4 operation comprises:

5 identifying one or more methods of the input component code that can be
6 called by the arbitrary code; and

7 identifying at least one method of the arbitrary code that can be called by a
8 virtual call of the identified one or more methods of the input component code.

9
10 40. The computer program product of claim 29 wherein any method
11 reachable by execution in accordance with the runtime security policy is
12 represented by one of more nodes in the call graph.

13 41. The computer program product of claim 29 wherein the generating
14 operation comprises:

15 generating at least one constraint associated with one or more instructions
16 in the input component code.

17
18 42. The computer program product of claim 29 wherein the generating
19 operation comprises:

20 generating at least one constraint associated with a parameter of a method
21 call in the input component code.

22
23
24
25

1 43. The computer program product of claim 29 wherein the generating
2 operation comprises:

3 generating at least one constraint associated with a returned result of a
4 method call in the input component code.

5 44. The computer program product of claim 29 wherein the computer
6 process further comprises:

7 analyzing the call graph to identify a call path that presents a security
8 vulnerability in the input component code.

9 45. The computer program product of claim 29 wherein the computer
10 process further comprises:

11 analyzing the call graph to identify a call path that presents a security
12 vulnerability in the input component code and a call path that presents no security
13 vulnerability in the input component code.

14 46. The computer program product of claim 29 wherein the computer
15 process further comprises:

16 analyzing the call graph to generate a security report that identifies a
17 security vulnerability in the input component code.

18 47. The computer program product of claim 29 wherein the computer
19 process further comprises:

20 analyzing the call graph to identify a call path that satisfies a query.
21
22
23
24
25

1 48. The computer program product of claim 29 wherein the computer
2 process further comprises:

3 analyzing the call graph to identify a security-vulnerable usage of a
4 permission demand.

5 49. The computer program product of claim 29 wherein the computer
6 process further comprises:

7 analyzing the call graph to identify a security-vulnerable usage of a
8 permission assertion.

9 50. The computer program product of claim 29 wherein the computer
10 process further comprises:

11 analyzing the call graph to identify a lack of uniform usage of security
12 checks.

13 51. The computer program product of claim 29 wherein the computer
14 process further comprises:

15 analyzing the call graph to identify an equivalence between use of a
16 permission link-demand and a permission demand. [Inventors: Can you define a
17 link-demand? Is this a Demand at link time? If so, I will use that definition
18 instead of the specific term link-demand.
19

20 52. The computer program product of claim 29 wherein the generating
21 operation comprises:

22 generating a class hierarchy that contains classes of the input component
23 code and symbolic classes that represent classes of the arbitrary code;
24
25

1 generating at least one constraint associated with a virtual call in the input
2 component code; and

3 evaluating the at least one constraint by a symbolic computation on
4 potential target classes for the virtual call in the generated class hierarchy.

5 53. The computer program product of claim 29 wherein the generating
6 operation comprises:

7 generating at least one constraint associated with either a security demand
8 or a security assert in the input component code;

9 evaluating the at least one constraint by a symbolic computation on
10 dynamic permissions of the input component code and on a parameter permission
11 of the security demand or the security assert; and

12 conditionally generating at least one additional constraint associated with
13 one or more instructions located in the input component code after the security
14 demand or assert, responsive to the evaluating operation.

15
16 54. The computer program product of claim 29 further comprising
17 analyzing the call graph to classify, based on permissions, pieces of code
18 performing sensitive actions in the input component code.

19 55. The computer program product of claim 29 further comprising
20 analyzing the call graph and another call graph obtained for a different
21 version of input component code to generate a security report that identifies a
22 security vulnerability in the different version of the input component code.

1 56. The computer program product of claim 29 further comprising
2 analyzing the call graph and another call graph obtained for a different
3 version of input component code to identify a call path that presents a security
4 vulnerability in the different version of the input component code.
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

1 57. A system comprising:

2 a call graph generator receiving into an execution environment input
3 component code and a runtime security policy, and generating a call graph of call
4 paths through the input component code simulated in combination with at least
5 one symbolic component that represents additional arbitrary code that complies
6 with the runtime security policy.
7

8 58. The system of claim 57 wherein a possible execution path through the
9 input component code that is compliant with the runtime security policy is
10 represented by an individual call path.

11 59. The system of claim 57 further comprising:

12 a call graph analyzer that analyzes the call graph to identify a security-
13 vulnerable usage of a permission demand.
14

15 60. The system of claim 57 further comprising:

16 a call graph analyzer that analyzes the call graph to identify a security-
17 vulnerable usage of a permission assertion.

18 61. The system of claim 57 further comprising:

19 a call graph analyzer that analyzes the call graph to identify a lack of
20 uniform usage of security checks.
21

22 62. The system of claim 57 further comprising:

23 a call graph analyzer that analyzes the call graph to identify an equivalence
24 between use of a permission link-demand and a permission demand.
25

1 63. A method comprising:
2 analyzing relative to at least one query a call graph of call paths through
3 input component code simulated in combination with at least one symbolic
4 component representing additional arbitrary code that complies with a runtime
5 security policy; and

6 identifying a subset of the call paths in the call graph that satisfy the query.
7

8 64. The method of claim 63 wherein the analyzing operation comprises:
9 analyzing the call graph to identify a security-vulnerable usage of a
10 permission demand.

11 65. The method of claim 63 wherein the analyzing operation comprises:
12 analyzing the call graph to identify a security-vulnerable usage of a
13 permission assertion.

14 66. The method of claim 63 wherein the analyzing operation comprises:
15 analyzing the call graph to identify a lack of uniform usage of security
16 checks.
17

18 67. The method of claim 63 wherein the analyzing operation comprises:
19 analyzing the call graph to identify an equivalence between use of a
20 permission link-demand and a permission demand.
21
22
23
24
25

1 68. A computer program product encoding a computer program for
2 executing on a computer system a computer process, the computer process
3 comprising:

4 analyzing relative to at least one query a call graph of call paths through
5 input component code simulated in combination with at least one symbolic
6 component representing additional arbitrary code that complies with a runtime
7 security policy; and

8 identifying a subset of the call paths in the call graph that satisfy the query.
9

10 69. The computer program product of claim 68 wherein the compute
11 process further comprises:

12 analyzing the call graph to identify a security-vulnerable usage of a
13 permission demand.

14 70. The computer program product of claim 68 wherein the compute
15 process further comprises:

16 analyzing the call graph to identify a security-vulnerable usage of a
17 permission assertion.

18 71. The computer program product of claim 68 wherein the compute
19 process further comprises:

20 analyzing the call graph to identify a lack of uniform usage of security
21 checks.
22
23
24
25

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

72. The computer program product of claim 68 wherein the compute
process further comprises:
analyzing the call graph to identify an equivalence between use of a
permission link-demand and a permission demand.

1 73. A system comprising:

2 a call graph analyzer analyzing relative to at least one query a call graph of
3 call paths through input component code simulated in combination with at least
4 one symbolic component representing additional arbitrary code that complies with
5 a runtime security policy, and identifying a subset of the call paths in the call
6 graph that satisfy the query.
7

8 74. The system of claim 73 wherein the call graph analyzer analyzes the call
9 graph to identify a security-vulnerable usage of a permission demand.

10 75. The system of claim 73 wherein the call graph analyzer analyzes the call
11 graph to identify a security-vulnerable usage of a permission assertion.
12

13 76. The system of claim 73 wherein the call graph analyzer analyzes the call
14 graph to identify a lack of uniform usage of security checks.

15 77. The system of claim 73 wherein the call graph analyzer analyzes the call
16 graph to identify an equivalence between use of a permission link-demand and a
17 permission demand.
18
19
20
21
22
23
24
25